Package ‘adverSCarial’

May 29, 2024

Title  adverSCarial, generate and analyze the vulnerability of scRNA-seq classifiers to adversarial attacks

Version  1.2.0

Description  adverSCarial is an R Package designed for generating and analyzing the vulnerability of scRNA-seq classifiers to adversarial attacks. The package is versatile and provides a format for integrating any type of classifier. It offers functions for studying and generating two types of attacks, single gene attack and max change attack. The single gene attack involves making a small modification to the input to alter the classification. The max change attack involves making a large modification to the input without changing its classification. The package provides a comprehensive solution for evaluating the robustness of scRNA-seq classifiers against adversarial attacks.

License  MIT + file LICENSE

Encoding  UTF-8

Roxygen  list(markdown = TRUE)

RoxygenNote  7.2.3

biocViews  Software, SingleCell, Transcriptomics, Classification

Suggests  knitr, RUnit, BiocGenerics, TENxPBMCData, CHETAH, stringr, LoomExperiment

Imports  gtools, S4Vectors, methods, DelayedArray

VignetteBuilder  knitr

git_url  https://git.bioconductor.org/packages/adverSCarial

git_branch  RELEASE_3_19

git_last_commit  0edb731

git_last_commit_date  2024-04-30

Repository  Bioconductor 3.19

Date/Publication  2024-05-29
advChar-class

Description

advChar is a class used to store the output values of the advMaxChange function. The result can be a vector of few thousands genes, so a specific show method is associated to this class to avoid overflooding the R scripts outputs.

Value

A advChar object

Examples

MyClassifier <- function(expr, clusters, target) {
  c("T cell", 0.9)
}

genes <- paste0("gene_",1:10000)
rna_expression <- data.frame(lapply(genes, function(x) numeric(0)))
rna_expression <- rbind(rna_expression, rep(1,10000))
rna_expression <- rbind(rna_expression, rep(2,10000))
colnames(rna_expression) <- genes
clusters_id <- c("B cell","T cell")
advGridMinChange

max_change_genes <- advMaxChange(rna_expression, clusters_id, "T cell", MyClassifier, advMethod="perc99")

max_change_genes

---

**advGridMinChange**

*Grid search of min change adversarial attack. Tries each combination on a cluster, given a list of genes and a list of modifications.*

**Description**

Grid search of min change adversarial attack. Tries each combination on a cluster, given a list of genes and a list of modifications.

**Usage**

```r
advGridMinChange(
  exprs,
  clusters,
  target,
  classifier,
  genes,
  modifications = list(c("perc1"), c("perc99")),
  returnFirstFound = FALSE,
  argForClassif = "DelayedMatrix",
  argForModif = "DelayedMatrix",
  verbose = FALSE,
  iamsure = FALSE
)
```

**Arguments**

- **exprs**: DelayedMatrix of numeric RNA expression, cells are rows and genes are columns - or a SingleCellExperiment object, a matrix or a data.frame. By default matrix and data.frame are converted to DelayedMatrix for memory performance, see `argForModif` argument for options.
- **clusters**: a character vector of the clusters to which the cells belong
- **target**: the name of the cluster to modify
- **classifier**: a classifier in the suitable format. A classifier function should be formatted as follow: classifier = function(expr, clusters, target) # Making the classification c("cell type", score)
  - **score**: should be numeric between 0 and 1, 1 being the highest confidence into the cell type classification. The matrix `expr` contains RNA expression values, the vector `clusters` consists of the cluster IDs for each cell in `expr`, and `target` is the ID of the cluster for which we want to have a classification. The function returns a vector with the classification result, and a score.
advGridMinChange

genes the character vector of genes to study
modifications the list of the modifications to study
returnFirstFound set to TRUE to return result when the first misclassification is found
argForClassif the type of the first argument to feed to the classifier function. 'DelayedMatrix' by default, can be 'SingleCellExperiment' or 'data.frame'.
argForModif type of matrix for the modification, 'DelayedMatrix' by default. Can be 'data.frame', which is faster, but need more memory.
verbose logical, set to TRUE to activate verbose mode
iamsure logical, prevents from expansive calculations when genes list is too long, set to TRUE to run anyway.

Details

This function aims to find the shortest combination of genes allowing to make a min change attack. It will test every possible combination for a given gene list. This function can take a long time to run, and we recommand to use the random walk search advRandWalkMinChange function instead for lists above 10 genes.

You can specify a list of modifications as so, each item of the list should be 1 or 2 length size. The 1 length vector must contain the prerecorded modifications, 'perc1' or 'perc99'. The 2 length vector must have as first item:

- 'fixed', in this case the second item should be the value to be replaced by.
- 'full_row_fct', 'target_row_fct', 'target_matrix_fct' or 'full_matrix_fct'. In this case the second item should be a function. Let’s say we want to analysis the susceptibility to min change attack for 3 modifications: "perc1", the modification of each value of the cluster by 1000, and a custom modification stored inside a function myFct. Then the 'modification' parameter should be: my_modifications = list(c("perc1"), c("fixed", 1000), c("full_matrix_fct", myFct))

Value

DataFrame results of the classification of all the grid combinations

Examples

library(DelayedArray)

MyClassifier <- function(expr, clusters, target) {
c("T cell", 0.9)
}

rna_expression <- DelayedArray(data.frame(CD4=c(0,0,0,0), CD8A=c(1,1,1,1), CD8B=c(2,2,3,3)))
genesc <- c("CD4", "CD8A")
clusters_id <- c("B cell", "T cell")

advGridMinChange(rna_expression, clusters_id, "T cell", MyClassifier, genes=genes, modifications = list(c("perc1"), c("perc99")))
myModif = function(x, y){
  return(sample(1:10,1))
}

my_modifications = list(c("perc1"),
  c("fixed", 1000),
  c("full_matrix_fct", myModif))
advGridMinChange(rna_expression, clusters_id, "T cell",
  MyClassifier, genes=genes, modifications = my_modifications)

advList-class  
adversCarial class

Description
advList is a class used to store the output values of the advSingleGene function. The result can be a list of few thousands genes:cell_type key/values, so a specific show method is associated to this class to avoid overflooding the R scripts outputs.

Value
A advList object

Examples
MyClassifier <- function(expr, clusters, target) {
  c("B cell", 0.9)
}

rna_expression <- data.frame(CD4=c(0,0,0,0), CD8A=c(1,1,1,1),
  CD8B=c(2,2,3,3))
genes <- c("CD4", "CD8A")
clusters_id <- c("B cell", "B cell", "T cell", "T cell")

adv_min_change <- advSingleGene(rna_expression, clusters_id, "T cell",
  MyClassifier, advMethod="perc99")

adv_min_change
advMaxChange

Find a max change adversarial attack. It finds the longer list of genes you can modify on a cluster without changing its classification.

Description

Find a max change adversarial attack. It finds the longer list of genes you can modify on a cluster without changing its classification.

Usage

advMaxChange(
  exprs,
  clusters,
  target,
  classifier,
  exclGenes = c(),
  genes = c(),
  advMethod = "perc99",
  advFixedValue = 3,
  advFct = NULL,
  maxSplitSize = 1,
  argForClassif = "DelayedMatrix",
  argForModif = "data.frame",
  verbose = FALSE
)

Arguments

exprs	DelayedMatrix of numeric RNA expression, cells are rows and genes are columns - or a SingleCellExperiment object, a matrix or a data.frame. By default, these are converted to a data.frame to increase speed performance during modifications. However, this conversion can consume a significant amount of memory, see 'argForModif' argument for options.

clusters	a character vector of the clusters to which the cells belong

target	the name of the cluster to modify

classifier	a classifier in the suitable format. A classifier function should be formatted as follow: classifier = function(expr, clusters, target) # Making the classification c("cell type", score)
score should be numeric between 0 and 1, 1 being the highest confidence into the cell type classification. The matrix expr contains RNA expression values, the vector clusters consists of the cluster IDs for each cell in expr, and target is the ID of the cluster for which we want to have a classification. The function returns a vector with the classification result, and a score.

exclGenes	a list of genes to exclude from the analysis
advMaxChange

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>genes</td>
<td>a list of genes in case you want to limit the attack on a subset of genes</td>
</tr>
<tr>
<td>advMethod</td>
<td>the name of the method to use</td>
</tr>
<tr>
<td>advFixedValue</td>
<td>the numeric value to use in case of advMethod=fixed</td>
</tr>
<tr>
<td>advFct</td>
<td>the function to use in case advMethod=full</td>
</tr>
<tr>
<td></td>
<td>target_row_fct, target_matrix_fct, full_matrix_fct</td>
</tr>
<tr>
<td>maxSplitSize</td>
<td>max size of dichotomic slices.</td>
</tr>
<tr>
<td>argForClassif</td>
<td>the type of the first argument to feed to the classifier function. 'DelayedMatrix' by default, can be 'SingleCellExperiment' or 'data.frame'.</td>
</tr>
<tr>
<td>argForModif</td>
<td>type of matrix during for the modification, 'DelayedMatrix' by default. Can be 'data.frame', which is faster, but need more memory.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical, set to TRUE to activate verbose mode</td>
</tr>
</tbody>
</table>

**Details**

This function aims to get the largest part of the genes that can be modified without altering the classification, considering a given modification. You can refer to the `advModifications` function documentation for more details on how to define a modification. The search is made by a dichotomic process, on a recursive function. At each iteration the function splits the genes in two groups. It proceeds to the modification of the RNA gene value of the first group, makes its classification. Then three possible scenarios:

- the classification is the same as the target cluster. We concat the genes list to the previous one, make the classification, and it still gives same classification. Then we return the genes list.
- the classification is the same as the target cluster. We concat the genes list to the previous one, make the classification, and it gives a different classification. This happens often, you can modify the gene A with a classification of T cell, or modify the gene B with a classification of T cell, but modifying A and B returns another classification. In this case we split the genes list in two and try again.
- the classification is not the same as the target cluster. In this case we split the genes list in two and try again. The iteration process stops when the length of the genes list is lower than the value of the `maxSplitSize` argument. So you should set it to 1 to have the maximum number of genes for the max change attack. This function is used by the `overMaxChange` function with a default argument value of 100 to increase speed, and still returns significant results.

**Value**

a character vector of genes you can modify on a cluster without modifying its classification

**Examples**

```r
library(DelayedArray)

MyClassifier <- function(expr, clusters, target) {
  c("T cell", 0.9)
}

rna_expression <- DelayedArray(data.frame(CD4=c(0,0,0,0), CD8A=c(1,1,1,1), CD8B=c(2,2,3,3)))
```

advModifications <- c("CD4", "CD8A")
clusters_id <- c("B cell","B cell","T cell","T cell")

advMaxChange(rna_expression, clusters_id,
             "T cell", MyClassifier, advMethod="perc99")

advModifications

Returns a modified RNA expression DelayedMatrix, or a modified SingleCellExperiment, for a
given cluster, for a given modification.

Description

Returns a modified RNA expression DelayedMatrix, or a modified SingleCellExperiment, for a
given cluster, for a given modification.

Usage

advModifications(
  exprs,
  genes,
  clusters,
  target,
  advMethod = "perc99",
  advFixedValue = 3,
  advFct = NULL,
  argForClassif = "DelayedMatrix",
  argForModif = "data.frame",
  verbose = FALSE
)

Arguments

exprs: DelayedMatrix of numeric RNA expression, cells are rows and genes are columns
- or a SingleCellExperiment object, a matrix or a data.frame. By default, these
are converted to a data.frame to increase speed performance during modifica-
tions. However, this conversion can consume a significant amount of memory,
see 'argForModif' argument for options.

genes: the character vector of genes to modify

clusters: a character vector of the clusters to which the cells belong

target: the name of the cluster to modify

advMethod: the name of the method to use

advFixedValue: the numeric value to use in case advMethod=fixed

advFct: the function to use in case advMethod belongs to the following list: full_row_fct,
target_row_fct, target_matrix_fct, full_matrix_fct

argForClassif: the type of the first argument to feed to the classifier function. 'DelayedMatrix'
by default, can be 'SingleCellExperiment' or 'data.frame'.
argForModif type of matrix during for the modification, 'DelayedMatrix' by default. Can be 'data.frame', which is faster, but need more memory.

verbose logical, set to TRUE to activate verbose mode

Details

The motivation for this function is to standardize the modifications we want to study in the attacks. We give as argument a DelayedMatrix of the RNA expression, the gene and the target cells we want to modify. Then we have three arguments allowing to specify what modification we want to apply on these cells. The advMethod contains, a specific prerecorded modification or an indication on how to use the other two arguments. The prerecorded modifications available for the advMethod argument are:

- 'perc1', replace the value by the whole matrix 1 percentile value of the gene. It is as if we biologically switched off the gene.
- 'perc99', replace the value by the whole matrix 99 percentile value of the gene. It is as if we biologically switched on the gene to the maximum.
- 'random', replace the value by from a uniform distribution between min and max of the gene on the dataset
- 'positive_aberrant' replace value by 10,000 times the max value of the gene on the dataset
- 'negative_aberrant' replace value by -10,000 times the max value of the gene on the dataset
- 'decile+X', shifts the gene value by + X deciles.
- 'decile-X', shifts the gene value by - X deciles. The value of the advMethod argument can also be 'fixed', in this case the modification would be to replace the value of the gene of the wanted cells by the value of the argument 'advFixedValue'. This can be useful to test aberrant values like negative integer, absurdly high values of character values. The value of the advMethod argument can also be 'full_row_fct', 'target_row_fct', 'target_matrix_fct' or 'full_matrix_fct'. They are used when we want to use a custom modification function, with the 'advFct' argument:
  - 'full_row_fct' indicate that the 'advFct' function takes the whole gene values as input.
  - 'target_row_fct' indicate that the 'advFct' function takes target cells gene values as input.
  - 'full_matrix_fct' indicate that the 'advFct' function takes the whole gene expression values as input.
  - 'target_matrix_fct' indicate that the 'advFct' function takes target cells all genes values as input.

Value

the matrix or a data.frame exprs modified on asked genes with the specified modification

Examples

library(DelayedArray)

rna_expression <- DelayedArray(data.frame(CD4=c(0,0,0,0), CD8A=c(1,1,1,1), CD8B=c(2,2,3,3)))
```r
genes <- c("CD4", "CD8A")
clusters_id <- c("B cell", "B cell", "T cell", "T cell")

advModifications(rna_expression, genes, clusters_id, "T cell", advMethod="perc99")
```

---

**advRandWalkMinChange**  
*Random walk search of min change adversarial attack.*

**Description**

Random walk search of min change adversarial attack.

**Usage**

```r
advRandWalkMinChange(
  exprs,
  clusters,
  target,
  classifier,
  genes,
  modifications = list(c("perc1"), c("perc99")),
  firstBatch = 100,
  walkLength = 100,
  stepChangeRatio = 0.2,
  whileMaxCount = 10000,
  changeType = "any",
  argForClassif = "DelayedMatrix",
  argForModif = "DelayedMatrix",
  verbose = FALSE
)
```

**Arguments**

- **exprs**  
  DelayedMatrix of numeric RNA expression, cells are rows and genes are columns  
  - or a SingleCellExperiment object, a matrix or a data.frame. By default matrix  
    and data.frame are converted to DelayedMatrix for memory performance, see  
    `argForModif` argument for options.

- **clusters**  
  a character vector of the clusters to which the cells belong

- **target**  
  the name of the cluster to modify

- **classifier**  
  a classifier in the suitable format. A classifier function should be formatted as  
  follow: classifier = function(expr, clusters, target) # Making the classification  
  c("cell type", score)  
  score should be numeric between 0 and 1, 1 being the highest confidence into  
  the cell type classification. The matrix expr contains RNA expression values,  
  the vector clusters consists of the cluster IDs for each cell in expr, and target  
  is the ID of the cluster for which we want to have a classification. The function  
  returns a vector with the classification result, and a score.
advRandWalkMinChange

**genes**  the character vector of genes to study

**modifications**  the list of the modifications to study

**firstBatch**  the maximum number of try in step 1

**walkLength**  the maximum number of try in step 2

**stepChangeRatio**  ratio of parameters change in new walk step

**whileMaxCount**  the maximum number of try when looking for new combination of parameters

**changeType**  any consider each misclassification, not_na consider each misclassification but NA.

**argForClassif**  the type of the first argument to feed to the classifier function. 'DelayedMatrix' by default, can be 'SingleCellExperiment' or 'data.frame'.

**argForModif**  type of matrix during for the modification, 'DelayedMatrix' by default. Can be 'data.frame', which is faster, but need more memory.

**verbose**  logical, set to TRUE to activate verbose mode

**Details**

The parameter search by grid can take a long time, this function aims to make a parameter search faster. We have a function, advSingleGene, looking for one gene attacks. The advRandWalkMinChange function aims to find a multiple genes attack, with the fewer genes possible. At first the user have to provide a list of genes to test, for example by running differential statistics between two cell clusters. The user should also provide a list of modifications to test, to define as so - each item of the list should be 1 or 2 length size. The 1 length vector must contain the prerecorded modifications, 'perc1' or 'perc99'. The 2 length vector must have as first item:

- 'fixed', in this case the second item should be the value to be replaced by.
- 'full_row_fct', 'target_row_fct', 'target_matrix_fct' or 'full_matrix_fct'. In this case the second item should be a function. Let's say we want to analysis the susceptibility to min change attack for 3 modifications: "perc1", the modification of each value of the cluster by 1000, and a custom modification stored inside a function myFct. Then the 'modification' parameter should be: my_modifications = list(c("perc1"), c("fixed", 1000), c("full_matrix_fct", myFct))

Then the function will try to find the best combination of these genes and modifications to make the min change attack. Step 1 is to find a seed by trying random combinations of genes and modifications on a cluster until the classification is altered. Step 2 is to perform a random walk search to reduce the number of genes needed to change the classification. The

**Value**

DataFrame results of the classification of all the grid combinations

**Examples**

```r
library(DelayedArray)

MyClassifier <- function(expr, clusters, target) {
  c("T cell", 0.9)
```

advSingleGene

Find a one gene min change adversarial attack list. A one gene min change adversarial attack refers to the modification of a single gene within a cluster, leading to a change in its classification. The function returns a list of genes/new classification.

Description

Find a one gene min change adversarial attack list. A one gene min change adversarial attack refers to the modification of a single gene within a cluster, leading to a change in its classification. The function returns a list of genes/new classification.

Usage

advSingleGene(
  exprs,
  clusters,
  target,
  classifier,
  exclGenes = c(),
  genes = c(),
  advMethod = "perc99",
  advFixedValue = 3,
advSingleGene

```r
advFct = NULL,
firstDichot = 100,
maxSplitSize = 1,
returnFirstFound = FALSE,
changeType = "any",
argForClassif = "DelayedMatrix",
argForModif = "data.frame",
verbose = FALSE
```

Arguments

**exprs**
- DelayedMatrix of numeric RNA expression, cells are rows and genes are columns
- or a SingleCellExperiment object, a matrix or a data.frame. By default, these are converted to a data.frame to increase speed performance during modifications. However, this conversion can consume a significant amount of memory, see ‘argForModif’ argument for options.

**clusters**
- a character vector of the clusters to which the cells belong

**target**
- the name of the cluster to modify

**classifier**
- a classifier in the suitable format. A classifier function should be formatted as follows: `classifier = function(expr, clusters, target)` # Making the classification
- `c("cell type", score)`
- score should be numeric between 0 and 1, 1 being the highest confidence into the cell type classification. The matrix `expr` contains RNA expression values, the vector `clusters` consists of the cluster IDs for each cell in `expr`, and `target` is the ID of the cluster for which we want to have a classification. The function returns a vector with the classification result, and a score.

**exclGenes**
- a character vector of genes to exclude from the analysis

**genes**
- a character vector of genes in case you want to limit the attack on a subset of genes

**advMethod**
- the name of the method to use

**advFixedValue**
- the numeric value to use in case of advMethod=fixed

**advFct**
- the function to use in case advMethod belongs to the following list: full_row_fct, target_row_fct, target_matrix_fct, full_matrix_fct

**firstDichot**
- the initial number of slices before the dichotomic search

**maxSplitSize**
- max size of dichotomic slices

**returnFirstFound**
- set to TRUE to return result when a first misclassification is found

**changeType**
- any consider each misclassification, not_na consider each misclassification but NA.

**argForClassif**
- the type of the first argument to feed to the classifier function. 'DelayedMatrix' by default, can be 'SingleCellExperiment' or 'data.frame'.

**argForModif**
- type of matrix during for the modification, 'DelayedMatrix' by default. Can be 'data.frame', which is faster, but need more memory.

**verbose**
- logical, set to TRUE to activate verbose mode
This function aims to get all genes that when modified individually can lead to a misclassification. You can refer to the 'advModifications' function documentation to more details on how to define a modification. The function is made as a two step parameter search. The first step is to split the genes in 'firstDichot' sets, 100 by default. Then each set is studied by a dichotomic process in a recursive function. The aim of sarting by a high value of sets, instead of starting directly by the dichotomic research is to avoid the following scenario: we modify 5000 genes, the modification of one gene compensates the modification of another. The classification remains unchanged, whereas there is a one gene classification modifying inside the 5000. The dichotomic process runs as follow. The function receives a list of genes, make the modification of the whole list and make the classification. Three scenarios possible:

- the classification remains the same as the target cluster. The function returns, and the dichotomic process continues.
- the classification is changed. There is only one gene in the list, the function returns the gene and the new classification.
- the classification is changed. There is more than one gene in the list, the genes list is split in two, and the dichotomic process continues.

Value

a list of genes/new classification tuples

Examples

library(DelayedArray)

MyClassifier <- function(expr, clusters, target) {
  c("B cell", 0.9)
}

rna_expression <- DelayedArray(data.frame(CD4=c(0,0,0,0), CD8A=c(1,1,1,1),
CD8B=c(2,2,3,3))
genes <- c("CD4", "CD8A")
clusters_id <- c("B cell","B cell","T cell","T cell")
advSingleGene(rna_expression, clusters_id, "T cell", MyClassifier, advMethod="perc99")

matrixFromSCE

Returns the RNA expression matrix from a SingleCellExperiment with unique hgnc gene names in columns

Description

Returns the RNA expression matrix from a SingleCellExperiment with unique hgnc gene names in columns
maxChangeOverview

Usage
matrixFromSCE(sce)

Arguments
sce SingleCellExperiment object to convert

Details
This function retrieves from a SingleCellExperiment object the raw RNA expression value corresponding to the hgnc genes. The resulting matrix can then be used with adverSCarial packages.

Value
the RNA expression matrix from a SingleCellExperiment with unique hgnc gene names in columns

Examples
library(TENxPBMCData)

pbmc <- TENxPBMCData(dataset = "pbmc3k")
mat_rna <- matrixFromSCE(pbmc)

maxChangeOverview

Given an overview of the susceptibility to max change attacks, for each cell type, for a given list of modifications.

Description
Gives an overview of the susceptibility to max change attacks, for each cell type, for a given list of modifications.

Usage
maxChangeOverview(
   exprs,
   clusters,
   classifier,
   exclGenes = c(),
   genes = c(),
   modifications = list(c("perc1"), c("perc99")),
   advMethod = "perc99",
   advFixedValue = 3,
   advFct = NULL,
   maxSplitSize = 100,
   argForClassif = "DelayedMatrix",
   argForModif = "data.frame",
)
maxChangeOverview

verbose = FALSE

Arguments

exprs
- a DelayedMatrix of numeric RNA expression, cells are rows and genes are columns. 
or a SingleCellExperiment object, a matrix or a data.frame. By default, these are converted to a data.frame to increase speed performance during modifications. However, this conversion can consume a significant amount of memory, see 'argForModif' argument for options.

clusters
- a character vector of the clusters to which the cells belong

classifier
- a classifier in the suitable format. A classifier function should be formatted as follow: classifier = function(expr, clusters, target) # Making the classification c("cell type", score)

score
- should be numeric between 0 and 1, 1 being the highest confidence into the cell type classification. The matrix expr contains RNA expression values, the vector clusters consists of the cluster IDs for each cell in expr, and target is the ID of the cluster for which we want to have a classification. The function returns a vector with the classification result, and a score.

exclGenes
- a character vector of genes to exclude from the analysis

genes
- a character vector of genes in case you want to limit the analysis on a subset of genes

modifications
- the list of the modifications to study

advMethod
- the name of the method to use

advFixedValue
- the numeric value to use in case advMethod=fixed

advFct
- the function to use in case advMethod belongs to the following list: full_row_fct, target_row_fct, target_matrix_fct, full_matrix_fct

maxSplitSize
- max size of dichotomic slices.

argForClassif
- the type of the first argument to feed to the classifier function. 'DelayedMatrix' by default, can be 'SingleCellExperiment' or 'data.frame'.

argForModif
- type of matrix during for the modification, 'DelayedMatrix' by default. Can be 'data.frame', which is faster, but need more memory.

verbose
- logical, set to TRUE to activate verbose mode

Details

Running the advMaxChange function for each cell type to see which ones are more vulnerable can take a long time. The aim of the maxChangeOverview function is to make this process faster. It uses a default value of 100 for the 'maxSplitSize' parameter. So, the dichotomic process of the advMaxChange function stops as soon as the fold length is lower than 100. You can have more accurate results with maxSplitSize=1, but it will take longer. This function aims also to run the advMaxChange for several given modifications. You can specify a list of modifications as so - each item of the list should be 1 or 2 length size. The 1 length vector must contain the prerecorded modifications, 'perc1' or 'perc99'. The 2 length vector must have as first item:

- 'fixed', in this case the second item should be the value to be replaced by.
• ‘full_row_fct’, ‘target_row_fct’, ‘target_matrix_fct’ or ‘full_matrix_fct’. In this case the second item should be a function. Let’s say we want to analysis the susceptibility to max change attack for 3 modifications: “perc1”, the modification of each value of the cluster by 1000, and a custom modification stored inside a function myFct. Then the ‘modification’ parameter should be: my_modifications = list(c("perc1"), c("fixed", 1000), c("full_matrix_fct", myFct))

The function returns a dataframe with the number of genes of the max change attack for each modification in columns, for each cell type in rows.

Value

A DataFrame storing the number of possible max change attacks for each cell type and each modification.

Examples

library(DelayedArray)

MyClassifier <- function(expr, clusters, target) {
  c("T cell", 0.9)
}
rna_expression <- DelayedArray(data.frame(CD4=c(0,0,0,0), CD8A=c(1,1,1,1),
                                             CD8B=c(2,2,3,3)))
genes <- c("CD4", "CD8A")
clusters_id <- c("B cell", "T cell")
maxChangeOverview(rna_expression, clusters_id,
                  MyClassifier, modifications = list(c("perc1"), c("perc99")))

myModif = function(x, y){
  return(sample(1:10,1))
}
my_modifications = list(c("perc1"),
                        c("fixed", 1000),
                        c("full_matrix_fct", myModif))
maxChangeOverview(rna_expression, clusters_id,
                  MyClassifier, modifications = my_modifications)

MClassifier

Example cell type classifier for pbmc clustered datasets.

Description

Example cell type classifier for pbmc clustered datasets.

Usage

MClassifier(exprs, clusters, target)
predictWithNewValue

Arguments

exprs DelayedMatrix of numeric RNA expression, cells are rows and genes are columns - or a SingleCellExperiment object, a matrix or a data.frame.
clusters vector of clusters to which each cell belongs

Details

This classifier aims at testing the adverSCarial package of real pbmc data. It is a simple marker based classifier. It looks at the average value of a few genes inside a cluster, and returns the associated cell type. Markers where found by differential expressions.

Value

a vector with the classification, and the odd

Examples

library(TENxPBMCData)

pbmc <- TENxPBMCData(dataset = "pbmc3k")
mat_rna <- matrixFromSCE(pbmc)
cell_types <- system.file("extdata","pbmc3k_cell_types.tsv", package = "adverSCarial")
cell_types <- read.table(cell_types, sep = "\t")$cell_type

MClassifier(mat_rna, cell_types, "DC")

predictWithNewValue Returns a classification and an odd value from a RNA expression DelayedMatrix or a SingleCellExperiment object, for given genes, for a given cluster, for a given modification.

Description

Returns a classification and an odd value from a RNA expression DelayedMatrix or a SingleCellExperiment object, for given genes, for a given cluster, for a given modification.

Usage

predictWithNewValue(exprs, genes, clusters,
predictWithNewValue

```r
predictWithNewValue = function(target,
                                classifier,
                                advMethod = "perc99",
                                advFixedValue = 3,
                                advFct = NULL,
                                argForClassif = "DelayedMatrix",
                                argForModif = "DelayedMatrix",
                                verbose = FALSE)
```

### Arguments

- **exprs**
  - DelayedMatrix of numeric RNA expression, cells are rows and genes are columns - or a SingleCellExperiment object, a matrix or a data.frame.

- **genes**
  - the character vector of genes to modify

- **clusters**
  - a character vector of the clusters to which the cells belong

- **target**
  - the name of the cluster to modify

- **classifier**
  - a classifier in the suitable format. A classifier function should be formated as follow:
    ```r
classifier = function(expr, clusters, target) # Making the classification
c("cell type", score)
```
  - score should be numeric between 0 and 1, 1 being the highest confidence into the cell type classification. The matrix `expr` contains RNA expression values, the vector `clusters` consists of the cluster IDs for each cell in `expr`, and `target` is the ID of the cluster for which we want to have a classification. The function returns a vector with the classification result, and a score.

- **advMethod**
  - the name of the method to use

- **advFixedValue**
  - the numeric value to use in case of `advMethod=fixed`

- **advFct**
  - the function to use in case `advMethod` belongs to the following list: full_row_fct, target_row_fct, target_matrix_fct, full_matrix_fct

- **argForClassif**
  - the type of the first argument to feed to the classifier function. 'DelayedMatrix' by default, can be 'SingleCellExperiment' or 'data.frame'.

- **argForModif**
  - type of matrix during for the modification, 'DelayedMatrix' by default. Can be 'data.frame', which is faster, but need more memory.

- **verbose**
  - logical, set to TRUE to activate verbose mode

### Details

This function aims to concatenate the following actions:

- modify the RNA gene expression
- classify the result

### Value

- a vector of the classification, and the associated odd
Examples

```r
library(DelayedArray)

MyClassifier <- function(expr, clusters, target) {
  c("T cell", 0.9)
}

rna_expression <- DelayedArray(data.frame(CD4=c(0,0,0,0),
                                         CD8A=c(1,1,1,1),
                                         CD8B=c(2,2,3,3)))

genes <- c("CD4", "CD8A")
clusters_id <- c("B cell","B cell","T cell","T cell")
predictWithNewValue(rna_expression, genes, clusters_id,
                    "T cell", MyClassifier, advMethod="perc99")
```

---

**sceConvertToHGNC**

*Returns a SingleCellExperiment object keeping unique HGNC gene*

Description

Returns a SingleCellExperiment object keeping unique HGNC gene

Usage

`sceConvertToHGNC(sce)`

Arguments

- `sce` SingleCellExperiment object to convert

Details

Sometimes classifiers need HGNC instead of ensemble genes to run. This function allows to make the conversion.

Value

the SingleCellExperiment object keeping unique HGNC gene

Examples

```r
library(TENxPBMData)

pbmc <- TENxPBMData(dataset = "pbmc3k")
hgnc_pbmc <- sceConvertToHGNC(pbmce)
```
**singleGeneOverview**

Gives an overview of the susceptibility to single gene attacks, for each cell type, for a given list of modifications.

### Description

Gives an overview of the susceptibility to single gene attacks, for each cell type, for a given list of modifications.

### Usage

```r
singleGeneOverview(
  exprs,
  clusters,
  classifier,
  exclGenes = c(),
  genes = c(),
  modifications = list(c("perc1"), c("perc99")),
  advMethod = "perc99",
  advFixedValue = 3,
  advFct = NULL,
  firstDichot = 100,
  maxSplitSize = 100,
  changeType = "any",
  argForClassif = "DelayedMatrix",
  argForModif = "data.frame",
  verbose = FALSE
)
```

### Arguments

- **exprs**: DelayedMatrix of numeric RNA expression, cells are rows and genes are columns - or a SingleCellExperiment object, a matrix or a data.frame. By default, these are converted to a data.frame to increase speed performance during modifications. However, this conversion can consume a significant amount of memory, see 'argForModif' argument for options.

- **clusters**: a character vector of the clusters to which the cells belong

- **classifier**: a classifier in the suitable format. A classifier function should be formatted as follow: classifier = function(expr, clusters, target) # Making the classification c("cell type", score)

- **exclGenes**: a character vector of genes to exclude from the analysis

- **genes**: a character vector of genes to include in the analysis

- **modifications**: a list of modifications to apply

- **advMethod**: the method to use for the adversarial attack

- **advFixedValue**: the fixed value to use for the adversarial attack

- **advFct**: the function to use for the adversarial attack

- **firstDichot**: the threshold for the first dichotomy

- **maxSplitSize**: the maximum size of the split

- **changeType**: the type of change to apply

- **argForClassif**: the argument for the classifier function

- **argForModif**: the argument for the modification function

- **verbose**: a boolean indicating whether to print verbose output
genes a character vector of genes in case you want to limit the analysis on a subset of genes
modifications the list of the modifications to study
advMethod the name of the method to use
advFixedValue the numeric value to use in case of advMethod=fixed
advFct the function to use in case advMethod belongs to the following list: full_row_fct, target_row_fct, target_matrix_fct, full_matrix_fct
firstDichot the initial number of slices before the dichotomic search
maxSplitSize max size of dichotomic slices.
changeType any consider each misclassification, not_na consider each misclassification but NA.
argForClassif the type of the first argument to feed to the classifier function. ‘DelayedMatrix’ by default, can be ‘SingleCellExperiment’ or ‘data.frame’.
argForModif type of matrix during for the modification, ‘DelayedMatrix’ by default. Can be ‘data.frame’, which is faster, but need more memory.
verbose logical, set to TRUE to activate verbose mode

Details

Running the advSingleGene function for each cell type to see which ones are more vulnerable can take a long time. The aim of the singleGeneOverview function is to make this process faster. It uses a default value of 100 for the ‘maxSplitSize’ parameter. So, the dichotomic process of the advSingleGene function stops as soon as the fold length is lower than 100. You can have more accurate results with maxSplitSize=1, but it will take longer. This function aims also to run the advSingleGene for several given modifications. You can specify a list of modifications as so - each item of the list should be 1 or 2 length size. The 1 length vector must contain the prerecorded modifications, ‘perc1’ or ‘perc99’. The 2 length vector must have as first item:

- ‘fixed’, in this case the second item should be the value to be replaced by.
- ‘full_row_fct’, ‘target_row_fct’, ‘target_matrix_fct’ or ‘full_matrix_fct’. In this case the second item should be a function. Let’s say we want to analysis the susceptibility to single gene attack for 3 modifications: “perc1”, the modification of each value of the cluster by 1000, and a custom modification stored inside a function myFct. Then the ‘modification’ parameter should be: my_modifications = list(c("perc1"), c("fixed", 1000), c("full_matrix_fct", myFct))

The function returns a dataframe with the number of genes of the max change attack for each modification in columns, for each cell type in rows.

Value

a DataFrame storing the number of possible single gene attacks each cell type and each modification.
Examples

```r
library(DelayedArray)

MyClassifier <- function(expr, clusters, target) {
  c("T cell", 0.9)
}

rna_expression <- DelayedArray(data.frame(CD4=c(0,0,0,0), CD8A=c(1,1,1,1),
                                        CD8B=c(2,2,3,3)))
genes <- c("CD4", "CD8A")
clusters_id <- c("B cell", "B cell", "T cell", "T cell")

singleGeneOverview(rna_expression, clusters_id,
                    MyClassifier, modifications = list(c("perc1"), c("perc99")))

myModif = function(x, y){
  return(sample(1:10,1))
}

my_modifications = list(c("perc1"),
                        c("fixed", 1000),
                        c("full_matrix_fct", myModif))

singleGeneOverview(rna_expression, clusters_id,
                    MyClassifier, modifications = my_modifications)
```
Index

advChar (advChar-class), 2
advChar-class, 2
advGridMinChange, 3
advList (advList-class), 5
advList-class, 5
advMaxChange, 6
advModifications, 8
advRandWalkMinChange, 10
advSingleGene, 12

matrixFromSCE, 14
maxChangeOverview, 15
MClassifier, 17

predictWithNewValue, 18

sceConvertToHGNC, 20
singleGeneOverview, 21